

HW2

黃佳溢

October 15, 2024

1 实验环境搭建

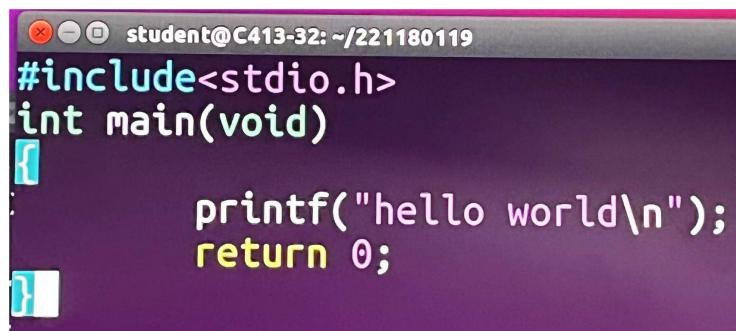
1.1 内核及文件系统加载

由于该部分和后面的内容有所交集，所以我们放在后面的章节展示。

1.2 交叉编译

由于实验室电脑是x86架构，而开发板使用龙芯架构，在开发时，我们需要使用龙芯架构的gcc编译器编译.c文件，生成的可执行文件才能在开发板上正常运行。

我们编写简易的hello world代码：



```
student@C413-32: ~/221180119
#include<stdio.h>
int main(void)
{
    printf("hello world\n");
    return 0;
}
```

Figure 1: Hello world

下面在进行编译时，不能直接使用 `gcc hello.c -o hello`，因为此时主机会使用x86架构的gcc编译，生成的可执行文件并不能被开发板所解析。为了解决这一点，我们需要使用交叉编译：

```
student@C413-32:~/221180119$ vi hello.c
student@C413-32:~/221180119$ ls
hello.c
student@C413-32:~/221180119$ export PATH=/opt/cross-tools/bin:$PATH
student@C413-32:~/221180119$ $PATH
bash: /opt/cross-tools/bin:/opt/cross-tools/bin:/opt/armhf-2018.09/bin:/home/stu
dent/bin:/home/student/.local/bin:/root/busybox-1.29.2/_install/bin:/usr/local/
sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/s
nap/bin: 没有那个文件或目录
student@C413-32:~/221180119$ loongarch64-linux-gnu-gcc hello -o hello.c
loongarch64-linux-gnu-gcc: error: hello: No such file or directory
loongarch64-linux-gnu-gcc: fatal error: no input files
compilation terminated.
student@C413-32:~/221180119$ loongarch64-linux-gnu-gcc hello.c -o hello
student@C413-32:~/221180119$ ls
hello hello.c
student@C413-32:~/221180119$ ./hello
bash: ./hello: cannot execute binary file: 可执行文件格式错误
student@C413-32:~/221180119$
```

Figure 2: Gcc

可以看到，此时编译的hello可执行文件不能被主机执行，当我们把hello转移到被挂载的nfs4文件并在开发板上执行，可以执行！

```
hel 221180119 # mv 221180119 221180119-1
one stu /mnt # mv 221180119-1 221180119
/mnt # ls
211180137
221180119
Makefile
Module.symvers
char_dev
cross-tools
cross-tools.tar.xz
extdir
hello
hi
jpeg
lcd
led
/mnt # ./hello
hello world
/mnt #
```

Figure 3: 开发板执行

2 内核和文件系统定制

2.1 默认内核制作

本次任务中，我们需要熟悉内核和文件系统的制作。首先我们先来制作内核vmlinuz。以下是按照给定流程所呈现的结果：

```
student@C413-16:~/221180119/linux_4.19.190.7.9$ make ARCH=loongarch CROSS_COMPILE=/opt/cross-tools/bin/loongarch64-linux-gnu-vmlinuxc
# configuration written to .config
#
#
```

Figure 4: 步骤一

```
student@C413-16:~/221180119/linux_4.19.190.7.9$ make -j8 ARCH=loongarch CROSS_COMPILE=/opt/cross-tools/bin/loongarch64-linux-gnu-vmlinuxc
MPILE=/opt/cross-tools/bin/loongarch64-linux-gnu-vmlinuxc: not found
./bin/sh: 1: /opt/cross-tools/bin/loongarch64-linux-gnu-vmlinuxc: not found
make[1]: *** [scripts/Makefile.build:10: vmlinux] Error 127
/bin/sh: 1: /opt/cross-tools/bin/loongarch64-linux-gnu-vmlinuxc: not found
./bin/sh: 1: /opt/cross-tools/bin/loongarch64-linux-gnu-vmlinuxc: not found
./bin/sh: 1: /opt/cross-tools/bin/loongarch64-linux-gnu-vmlinuxc: not found
scripts/kconfig/conf -syncconfig Kconfig
./scripts/gcc-version.sh: ./scripts/gcc-version.sh: /opt/cross-tools/bin/loongarch64-linux-gnu-vmlinuxgcc: not found
./scripts/gcc-version.sh: 27: ./scripts/gcc-version.sh: /opt/cross-tools/bin/loongarch64-linux-gnu-vmlinuxgcc: not found
./scripts/gcc-version.sh: 29: ./scripts/gcc-version.sh: /opt/cross-tools/bin/loongarch64-linux-gnu-vmlinuxgcc: not found
./scripts/gcc-version.sh: 26: ./scripts/gcc-version.sh: /opt/cross-tools/bin/loongarch64-linux-gnu-vmlinuxgcc: not found
./scripts/gcc-version.sh: 26: ./scripts/gcc-version.sh: /opt/cross-tools/bin/loongarch64-linux-gnu-vmlinuxgcc: not found
```

Figure 5: 步骤二

```
student@C413-16:~/221180119/linux_4.19.190.7.9$ make -j8 ARCH=loongarch CROSS_COMPILE=/opt/cross-tools/bin/loongarch64-linux-gnu-
modules_install INSTALL_MOD_PATH=/tmp/modules
INSTALL block/deadline-iosched.ko
INSTALL crypto/aubits.ko
INSTALL crypto/arc4.ko
INSTALL crypto/asym_tx/async_nemcpy.ko
INSTALL crypto/asym_tx/async_pg_ko
INSTALL crypto/asym_tx/async_ratd_recov.ko
INSTALL crypto/asym_tx/async_xor_ko
INSTALL crypto/authenc.ko
INSTALL crypto/async_tx/async_tx.ko
INSTALL crypto/bf/cbc.ko
INSTALL crypto/blowfish/common.ko
INSTALL crypto/blowfish/generic.ko
INSTALL crypto/cast5/cbc.ko
INSTALL crypto/cast6/generic.ko
INSTALL crypto/cast_common.ko
INSTALL crypto/cast_ko
INSTALL crypto/castdeflate.ko
INSTALL crypto/ccm.ko
INSTALL crypto/des_generic.ko
```

Figure 6: 步骤三

```
student@C413-16:~/221180119/linux_4.19.190.7.9$ make -j8 ARCH=loongarch CROSS_COMPILE=/opt/cross-tools/bin/loongarch64-linux-gnu-
modules_install INSTALL_MOD_PATH=/tmp/modules
INSTALL block/deadline-iosched.ko
INSTALL crypto/aubits.ko
INSTALL crypto/arc4.ko
INSTALL crypto/asym_tx/async_nemcpy.ko
INSTALL crypto/asym_tx/async_pg_ko
INSTALL crypto/asym_tx/async_ratd_recov.ko
INSTALL crypto/asym_tx/async_xor_ko
INSTALL crypto/authenc.ko
INSTALL crypto/async_tx/async_tx.ko
INSTALL crypto/bf/cbc.ko
INSTALL crypto/blowfish/common.ko
INSTALL crypto/blowfish/generic.ko
INSTALL crypto/cast5/cbc.ko
INSTALL crypto/cast6/generic.ko
INSTALL crypto/cast_common.ko
INSTALL crypto/cast_ko
INSTALL crypto/castdeflate.ko
INSTALL crypto/ccm.ko
INSTALL crypto/des_generic.ko
```

Figure 7: 步骤四

这里简单解释一下命令行：

1 由于linux_4.19.190.7.9文件可以制作多种内核，并不仅限于龙芯。所以在执行make命令时，需要注明ARCH=loongarch

2 我们执行第一步指令后，Makefile文件会将生成相应的配置文件.config

3 第二三四步指令是采用龙芯架构的gcc编译器进行编译时，需要指定编译器前缀，于是我们需要指定CROSS_COMPILE。执行make vmlinuz、modules和modules_install

完成这些之后，我们采用tftp协议，将生成的内核文件vmlinuz放置在/srv/tftpboot下，并在开发板中下载并加载，发现无法进入交互页面：

```

[ 3.616473] Stack : 90000000001016de8 9000000000c7bde4 90000000dc190000 900000
[ 3.624492] 0000000000000000 90000000dc193b80 0000000000000000 900000
[ 3.632589] 900000000012f5e9f 900000000812f5e97 0000000000000010 000004
[ 3.640525] 9000000000c7b6e4 900000000010b5ear 0000000000000001 000008
[ 3.648541] 90000000008454f0 0000000000000000 00000000000000168 00000f
[ 3.656556] 000000000009397e 0000000000410c000 9000000000124c0c8 000000
[ 3.664572] 90000000001196c48 0000000000000000 0000000000000000 900000
[ 3.672587] fffffffffffffa ffffffffffffffa3 ffffffffffffeaf 900008
[ 3.680602] 9000000000209874 0000000000000000 000000000000000b 000004
[ 3.688618] 0000000000000000 0000000000071fff 000000000001800 900008
[ 3.696634] ...
[ 3.699078] Call Trace:
[ 3.701535] [<90000000000209874>] show_stack+0x34/0x140
[ 3.706683] [<90000000000c7b6e4>] dump_stack+0xa8/0xdc
[ 3.711744] [<90000000000c75f58>] panic+0x124/0x278
[ 3.716541] [<900000000011ed4e4>] mount_block_root+0x2cc/0x38c
[ 3.722293] [<900000000011ed7a8>] prepare_namespace+0x15c/0x1a4
[ 3.728138] [<900000000011ecfc4>] kernel_init_freeable+0x368/0x3a0
[ 3.734238] [<90000000000c7d0cc>] kernel_init+0x18/0xfc
[ 3.739381] [<9000000000020300c>] ret_from_kernel_thread+0xc/0x10
[ 3.745407] ---[ end Kernel panic - not syncing: VFS: Unable to mount root f-
[ 51.331771] random: crng init done
NOT Shut down slave cores

```

Figure 8: 加载vmlinuz

这是正常的，注意看倒数第二行的提示：**VFS: Unable to mount root f-**，这代表着我们缺少文件系统，无法建立unix系统的虚拟文件系统，也就无法进入交互页面。

2.2 定制内核制作

为了搭配定制文件系统，我们需要对内核的默认配置进行更改，由于本部分按照课件老老实实运行，基本没有出现问题，并且很多命令的解释都非常详实，这里我便只贴出部分步骤图片：

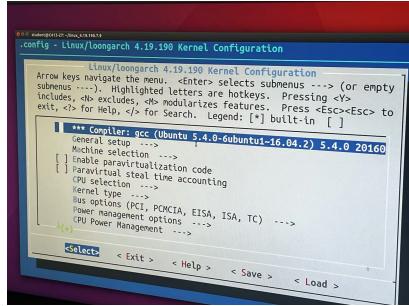


Figure 9: 步骤一

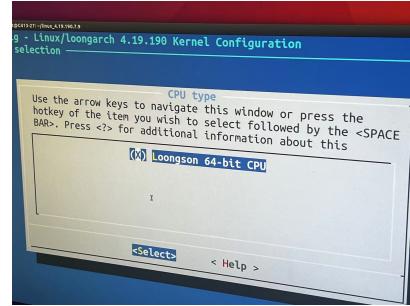


Figure 10: 步骤二

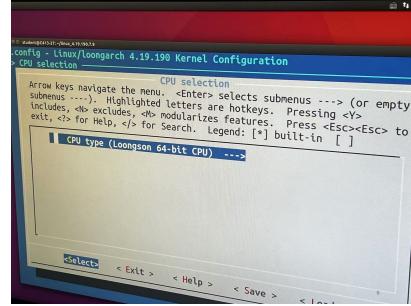


Figure 11: 步骤三

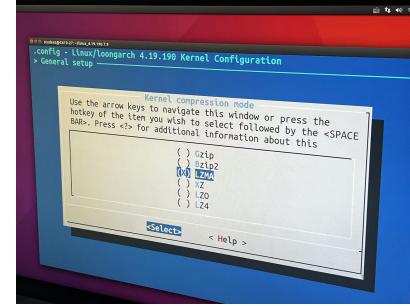


Figure 12: 步骤四

完成相关配置，执行`ARCH=loongarch CROSS_COMPILE=loongarch64-linux-gnu- make -j8 vmlinuz`，编译内核文件，完成。

2.3 文件系统定制

这个部分是最卡住我们的部分，有很多踩坑的地方，会在后面的章节进行介绍。这里还是贴出我们的执行图片：

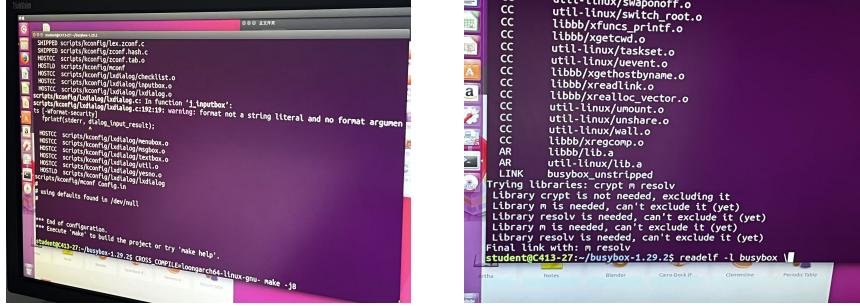


Figure 13: 步骤一

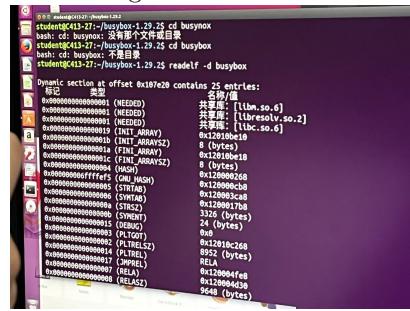


Figure 15: 步骤三

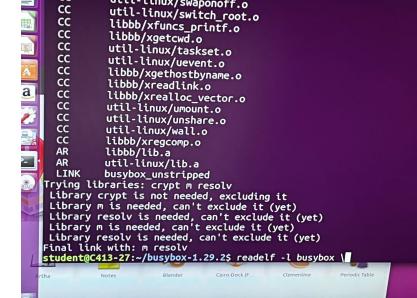


Figure 14: 步骤二



Figure 16: 步骤四

这里也就点一些注意事项:

1. 当我们需要链接共享库时，我们可以在某个可能的目录中执行**find -name ld-2.28.so**，该指令会告诉你哪里有。最好建议不要在根目录执行该指令，不然你会等它搜索半天。
2. 不要在`.install`目录下生成`ramdisk.img`，最好在`home`目录下，否则在后续的操作中，可能会出现递归套用的风险，因为你此时会将`.install`中的所有东西都拷贝进`ramdisk.img`里面，包括`ramdisk.img`本身。
3. 在操作中，我们需要将初始化好的`ramdisk.img`挂载到`/mnt/ramdisk`目录下，可能会出现报错，此时可以查看自己主机的`/etc/fstab`文件，看看有没有一行:
`ramdisk.img /mnt/ramdisk auto noauto ,users ,loop 0 0`
 没有就去找老师开权限写。

差不多注意这些点，老老实实按照课件，效率会高一些。尽管如此，我们周围还是很多人会在最后开发板加载时失败（包括我们），原因也五花八门，这在下一章提及。

2.4 践坑

1. make ARCh=loongarch loongson 22k1000 defconfig报错

```
student@C413-16:~/221180119/linux_4.19.190.7.9$ make ARCH=loongarch loongson_22k1000_defconfig
HOSTCC scripts/basic/fixed
make[1]: *** No rule to make target 'loongson_22k1000_defconfig'. 停止。
Makefile:556: recipe for target 'loongson_22k1000_defconfig' failed
make: *** [Loongson_22k1000_defconfig] Error 2
student@C413-16:~/221180119/linux_4.19.190.7.9$ make ARCH=loongarch loongson_22k1000_defconfig
HOSTCC scripts/kconfig/conf.o
YACC scripts/kconfig/zconf.tab.c
LEX scripts/kconfig/zconf.lex.c
HOSTCC scripts/kconfig/zconf.tab.o
HOSTLD scripts/kconfig/conf
*** Can't find default configuration "arch/loongarch/configs/loongson_22k1000_defconfig"!
scripts/kconfig/Makefile:104: recipe for target 'loongson_22k1000_defconfig' failed
make[1]: *** [Loongson_22k1000_defconfig] Error 1
Makefile:556: recipe for target 'loongson_22k1000_defconfig' failed
make: *** [Loongson_22k1000_defconfig] Error 2
student@C413-16:~/221180119/linux_4.19.190.7.9$
```

Figure 17: 践坑一

解决方法：查看报错信息，发现是系统无法找到22k1000 defconfig文件，于是我们跟随报错路径进行查看：

```
student@C413-16:~/221180119/linux_4.19.190.7.9/arch/loongarch/configs$ ls
loongson2_defconfig      loongson3_defconfig
loongson_2k1000_defconfig loongson3_server_defconfig
loongson_2k1500_defconfig
student@C413-16:~/221180119/linux_4.19.190.7.9/arch/loongarch/configs$
```

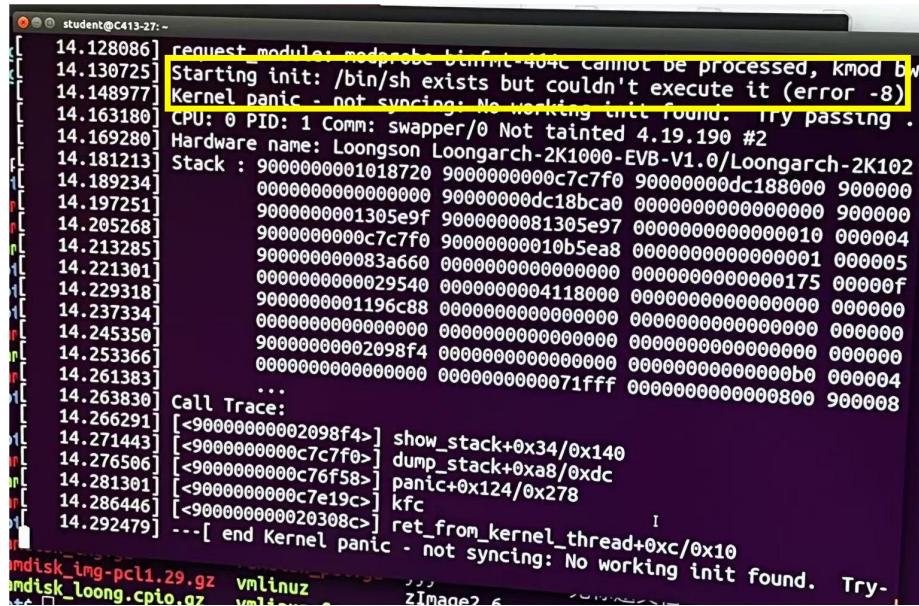
发现此时并没有22k1000 defconfig文件，也就怪不得无法执行，此时将命令修改为make ARCh=loongarch loongson 2k1000 defconfig即可。

2. 第二条指令中， tools打成了toools

3. 第三条指令中， gnu-和vmlinuz之间有空格，当时没注意，连在一起，导致无法寻找到该前缀！

4. 开发板加载定制内核、文件系统报错

这个是一个让我们很头疼的事情，但幸运的是，我们的定制系统从头到尾只有一个bug，解决掉也就能正常运行了。下面分享一下，该bug的基本特征，以及如何识别并解决。



```
student@C413-27: ~
[ 14.128086] request_module: modprobe binfmt-40c cannot be processed, kmod bw
[ 14.130725] Starting init: /bin/sh exists but couldn't execute it (error -8)
[ 14.148977] Kernel panic - not syncing: No working init found. Try passing .
[ 14.163180] CPU: 0 PID: 1 Comm: swapper/0 Not tainted 4.19.190 #2
[ 14.169280] Hardware name: Loongson Loongarch-2K1000-EVB-V1.0/Loongarch-2K102
[ 14.181213] Stack : 90000000001018720 9000000000c7c7f0 90000000dc188000 900000
[ 14.189234] 0000000000000000 90000000dc18bca0 0000000000000000 900000
[ 14.197251] 9000000001305e9f 90000000081305e97 0000000000000010 000004
[ 14.205268] 9000000000c7c7f0 90000000010b5ea8 0000000000000001 000005
[ 14.213285] 900000000083a660 0000000000000000 00000000000000175 00000f
[ 14.221301] 0000000000029540 0000000004118000 0000000000000000 000000
[ 14.229318] 9000000001196c88 0000000000000000 0000000000000000 000000
[ 14.237334] 0000000000000000 0000000000000000 0000000000000000 000000
[ 14.245350] 90000000002098f4 0000000000000000 0000000000000000 000000
[ 14.253366] 0000000000000000 0000000000000000 0000000000000000 000004
[ 14.261383] ...
[ 14.263830] Call Trace:
[ 14.266291] <90000000002098f4> show_stack+0x34/0x140
[ 14.271443] <9000000000c7c7f0> dump_stack+0xa8/0xdc
[ 14.276506] <9000000000c76f58> panic+0x124/0x278
[ 14.281301] <9000000000c7e19c> kfc
[ 14.286446] <900000000020308c> ret_from_kernel_thread+0xc/0x10
[ 14.292479] ...[ end Kernel panic - not syncing: No working init found. Try-
ramdisk_img-pcl1.29.gz vmlinuz   ...
ramdisk_loong.cpio.gz vmlinux   ...
zImage2   ...
zImage   ...
]
I
```

Figure 18: bug

很多时候我们不要被下面的**Kernel panic**吓唬到，需要往前找，注意到我们标黄色框框的地方，**/bin/sh exists but couldn't execute it**。

这种情况代表着你的文件系统可能不是**龙芯架构**。判断的方法便是：进入到`.install/bin`中，找到`busybox`文件，执行`file busybox`，就可以看出到底是什么架构的，一般来说，显示**unknown arch**应该就没啥问题。

假如和我们当时一样，显示**x86 arch**，那就是前面在定制文件系统时，编译或者下载的某一步没有加上**ARCH=loongarch**导致的，建议你删掉做好的`ramdisk.img`，重新按照教程，老老实实地做一次。

3 总结致谢

最后，我得说，大多数同学都会卡在最后一步无法进入交互界面，我还看到别的同学那种，没有ram0等等各种各样的bug，他们最终是否解决，如何解决都不得而知，遇到问题还是多去找老师，少红温，放平心态才是最重要的！

最后的最后，感谢我的好舍友吕展航和尤智宣同学对我的帮助和支持！感谢实验室的老师和助教们！